A Projection-Stable Grammatical Model to Specify Workflows for their P2P and Artifact-Centric Execution

Milliam Maxime Zekeng Ndadji, Maurice Tchoupé Tchendji, Didier Parigot

Rubrique

A Projection-Stable Grammatical Model to Specify Workflows for their P2P and Artifact-Centric Execution

Milliam Maxime ZEKENG NDADJI[1,a], Maurice TCHOUPÉ TCHENDJI[1,b], Didier PARIGOT[2,c]

[1]Department of Mathematics and Computer Science
University of Dschang, PO Box 67, Dschang-Cameroon
[2]Inria, Sophia Antipolis, France
[a]ndadji.maxime@univ-dschang.org, ndadjimaxime@yahoo.fr
[b]maurice.tchoupe@univ-dschang.org, ttchoupe@yahoo.fr
[c]didier.parigot@inria.fr

ABSTRACT. In this manuscript, we are interested in the specification and decentralized execution of administrative workflows. We present a grammatical model to specify such processes by indicating, in addition to their fundamental elements, the permissions (reading, writing and execution) of each actor in relation to each of the tasks that compose them. We then present a decentralized and artifact-centric execution model of these processes, on a Peer to Peer (P2P) Workflow Management System (WfMS). Our execution model allows the confidential execution of certain tasks by ensuring that, each actor potentially has only a partial perception of the overall process execution status. In our approach, we propose various stable projection algorithms, making it possible to obtain, to verify the coherence and to guarantee the convergence of the various potentially partial perceptions. Our algorithms are then coded and tested using a graphical tool to simulate the decentralized execution of administrative processes.

RÉSUMÉ. Dans ce manuscrit, nous nous intéressons à la spécification et à l'exécution décentralisée des processus workflows administratifs. Nous présentons un modèle grammatical permettant de spécifier de tels processus en précisant en plus de leurs éléments fondamentaux, les permissions (lecture, écriture et exécution) de chaque acteur relativement à chacune des tâches qui les composent. Nous présentons ensuite un modèle d'exécution décentralisée et centré-artefact de ces processus, sur un Système de Gestion de Workflows (SGWf) Pair à Pair (P2P). Notre modèle d'exécution autorise l'exécution confidentielle de certaines tâches en permettant à chaque acteur, de n'avoir potentiellement qu'une perception partielle de l'état d'exécution global du processus. Dans notre démarche, nous proposons divers algorithmes de projection stables, permettant d'obtenir, de vérifier la cohérence et de garantir la convergence des diverses perceptions potentiellement partielles. Nos algorithmes sont ensuite codés et expérimentés à l'aide d'un outil graphique de simulation d'exécution décentralisée des processus administratifs.

KEYWORDS : Workflow Management Systems, Workflow Specification Language, Artifacts, Accreditation, Projection, Partial replica, Peer to Peer.

MOTS-CLÉS : Système de Gestion des Workflows, Langage de Spécification des Workflows, Artefacts, Accréditation, Projection, Réplique Partielle, Pair à Pair.

1. Introduction

To easily automate their business processes and increase their competitiveness, organisations are increasingly interested in workflow technology. Indeed, in its most widespread approach, the latter reduces the automation of a given business process to its formal specification (modeling) using a *Workflow Process Specification Language* (WfPSL) [1, 3].

To automate a given administrative process [1], we can conceive it as a set of annotated trees called *target artifacts*, representing possible execution scenarios of the said process (see fig. 1). From this set of artifacts, we will derive an equivalent grammar called *Grammatical Model of Workflow* (GMWf). By associating this model with a list (subsequently called *set of accreditations*) informing on the permissions of each process actor for each task, we obtain a complete (ready to be executed) specification of the studied process.

Instances of an administrative process described using a GMWf and a set of accreditations, can be executed in a decentralized mode, by a *P2P-WfMS-View*; it is a *Workflow Management System* (WfMS) whose geographically dispersed instances (peers) communicate in Peer to Peer (P2P) by exchanging artifacts (*artifact-centric*). As a prerequisite to do so, each peer is configured using the model (GMWf + set of accreditations) of the process. From the (global) GMWf, each peer completes its configuration by deriving by *projection*, a local GMWf according to the accreditations of the local actor (*GMWf projection*). During the proper execution, actors coordinate by exchanging locally updated artifacts, to incrementally build one of the target artifacts: that's why the workflow execution process is said to be artifact-centric. On a given peer, when an artifact is received, it is projected (*artifact projection*) in accordance with the accreditations of the local actor in order to meet any confidentiality requirements. The local actor potentially manipulates only a partial replica of the global artifact under execution; his actions on it are guided by his local GMWf. At the end of its local execution, his updates to the manipulated partial replica must be integrated into the global artifact being executed. We then merge these two artifacts that are conform to two different models (we say that we realize the *expansion-pruning* of the updated partial replica). When this merging is complete, the process execution continues with the updated global artifact under execution.

Contributions: In this paper, we propose 1) a simple *Language for Specifying Administrative Workflow Processes* (LSAWfP), allowing to model each administrative process in the form of a GMWf to which a set of actors accreditations is added; 2) stable projection operations (*GMWf projection*, *artifacts projection* and *expansion-pruning*) for decentralized execution of processes using a P2P-WfMS-View.

Manuscript organization: in section 2 of this manuscript, we give a formal definition of our language (LSAWfP) and then we discuss its expressiveness. Then, in section 3, we present our model for decentralized execution of administrative processes specified using our LSAWfP, as well as its key algorithms. Finally, in appendices, we analyze the stability of our algorithms (sec. A) and we present a prototype of a P2P WfMS integrating our algorithms.

1. These are variable processes of which we know all the cases; that is, the tasks are predictable and the sequence are simple and clearly defined [8].

2. A Language for the Specification of Administrative Workflow Processes (LSAWfP)

2.1. Language definition

In the rest of this manuscript, we designate by the expression "*a Language for the Specification of Administrative Workflow Processes*" (LSAWfP), the language that we propose for processes specification. Also, we refer to *Grammatical Model of Administrative Workflow Process* (GMAWfP) as any specification of a business process produced using the language LSAWfP. Formally, a GMAWfP is defined as follows :

Définition 1 *The **Grammatical Model of Administrative Workflow Process** (GMAWfP) $\mathbb{W}_f$ of a given business process, is a triplet $\mathbb{W}_f = \left(\mathbb{G}, \mathcal{L}_{\mathcal{P}_k}, \mathcal{L}_{\mathcal{A}_k}\right)$ wherein $\mathbb{G}$ is the studied process (global) GMWf, $\mathcal{L}_{\mathcal{P}_k}$ is the set of k actors taking part in its execution and $\mathcal{L}_{\mathcal{A}_k}$ represents the set of these actors accreditations.*

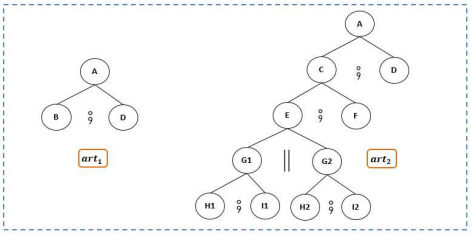

Figure 1. *Example of target artifacts for a given process (peer-review process)*

2.1.1. Concept of GMWf

The set of tasks of an administrative process and their ordering can be described using a set of *target artifacts* (see fig. 1). In these, nodes represent tasks to be executed and each hierarchical decomposition (a node and its sons) represents an ordering. In the artifacts that we handle, we consider only two types of ordering : (1) *sequential ordering*, noted $X_0 \rightarrow X_{s1} \fatsemi \ldots \fatsemi X_{sn}$ for a set $\{X_0, X_{s1} \ldots X_{sn}\}$ of tasks, which describes how the task X_0 precedes tasks $X_{s1} \ldots X_{sn}$ that should be executed sequentially and, (2) *parallel ordering*, noted $X_0 \rightarrow X_{p1} \parallel \ldots \parallel X_{pn}$, which indicates how the task X_0 precedes tasks $X_{p1}, \ldots, X_{pn}$ that can be executed concurrently. The set of target artifacts can therefore be substituted by a grammar $\mathbb{G}$ (a GMWf) in which, each symbol refers to a task and, each production p is of one of the following two forms: $p : X_0 \rightarrow X_1 \fatsemi \ldots \fatsemi X_n$ or $p : X_0 \rightarrow X_1 \parallel \ldots \parallel X_n$. In this case, each target artifact t_i *is conform to* $\mathbb{G}$ and we note $t_i \therefore \mathbb{G}$. More formally,

Définition 2 *A **Grammatical Model of Workflow** (GMWf) is defined by $\mathbb{G} = (\mathcal{S}, \mathcal{P}, A)$ where $\mathcal{S}$ is a finite set of **grammatical symbols** or **sorts** corresponding to various **tasks** to be executed in the studied business process; $A \in \mathcal{S}$ is a particular symbol called **axiom**, and $\mathcal{P} \subseteq \mathcal{S} \times \mathcal{S}^*$ is a finite set of **productions** decorated by the operators "$\fatsemi$" (is sequential to) and "$\parallel$" (is parallel to): they are **precedence rules**. A production $P = \left(X_{P(0)}, X_{P(1)} \cdots X_{P(|P|)}\right)$ is either of the form $P : X_0 \rightarrow X_1 \fatsemi \ldots \fatsemi X_{|P|}$, or of the form $P : X_0 \rightarrow X_1 \parallel \ldots \parallel X_{|P|}$ and $|P|$ designates the length of P right-hand side. Each grammatical symbol $X \in \mathcal{S}$ is associated with an attribute called **status**, that can be updated when the task X is executed; **X.status** provides access (read and write) to its content. A production with the symbol X as left-hand side is called a X-production.*

For some business processes, there may be particular cases where it is impossible to strictly order all tasks using the two production forms adopted for GMWf [2]. In such case, it is neccessary to insert new symbols, known as *(re)structuring symbols*, not related to tasks, to achieve a correct ordering that meets the forms of productions. To take into account such cases, we adapt the previously made GMWf definition to obtain the following complete one:

Définition 3 *A **Grammatical Model of Workflow** (GMWf) is defined by* $\mathbb{G} = (\mathcal{S}, \mathcal{P}, A_{\mathbb{G}})$ *wherein* $\mathcal{P}$ *refers to the same purpose as in definition 2,* $\mathcal{S} = \{A_{\mathbb{G}}\} \cup \mathcal{T} \cup \mathcal{T}_{Struc}$ *is a finite set of **grammatical symbols** or **sorts** in which, those of* $\mathcal{T}$ *correspond to **tasks** of the studied business process, while those of* $\mathcal{T}_{Struc}$ *are (re)structuring symbols.* $A_{\mathbb{G}}$ *is the axiom of grammar.*

2.1.2. Concept of accreditation of an actor

Since business processes are generally carried out collectively, it is necessary for coordination and confidentiality reasons, to clarify the permissions of each actor on each process task. In our case, we propose to do this when specifying the process, using accreditations. The accreditation of a given actor provides information on the rights (permissions) he has on each sort (task) of the GMWf. The nomenclature of the rights we handle is inspired by the one used in Unix-like operating systems. Three types of accreditation are therefore defined: accreditation in reading *(r)*, in writing *(w)* and in execution *(x)*.

1. *Accreditation in reading (r)*: an actor accredited in reading on sort X, may be not only informed on the execution of the related task, but he may also access its execution status. We call an actor's ***vue*** the set of sorts on which he is accredited in reading.

2. *Accreditation in writing (w)*: an actor accredited in writing on sort X is in charge of the execution of the related task. Any actor accredited in writing on a symbol must necessarily be accredited in reading on it.

3. *Accreditation in execution (x)*: an actor accredited in execution on sort X is authorized to ask to the actor who is accredited in writing on it to execute it.

More formally, an accreditation is defined as follows:

Définition 4 *An **accreditation** $\mathcal{A}_{A_i}$ defined on the set $\mathcal{S}$ of grammatical symbols for an actor A_i, is a triplet $\mathcal{A}_{A_i} = \left(\mathcal{A}_{A_i(r)}, \mathcal{A}_{A_i(w)}, \mathcal{A}_{A_i(x)}\right)$ such that, $\mathcal{A}_{A_i(r)} \subseteq \mathcal{S}$ also called **view** of actor A_i, is the set of symbols on which A_i is accredited in reading, $\mathcal{A}_{A_i(w)} \subseteq \mathcal{A}_{A_i(r)}$ is the set of symbols on which A_i is accredited in writing and $\mathcal{A}_{A_i(x)} \subseteq \mathcal{S}$ is the set of symbols on which A_i is accredited in execution.*

2.2. On the expressiveness of our language

Our language (LSAWfP) covers the three workflow conceptual models identified in [4]. In our case, the *organizational model* (that expresses and assigns the resources that must execute tasks) is specified using the couple $\left(\mathcal{L}_{\mathcal{P}_k}, \mathcal{L}_{\mathcal{A}_k}\right)$. As for the *information model* (it describes the data structure being manipulated), it is given by the type of the attribute *status* [3]. LSAWfP does not impose any constraints on the type of this attribute and leaves

2. For example, this is true for a process with four tasks A, B, C et D such as: A precedes all the others, B and C can be concurrently executed and they precede D. Here, the insertion of a new symbol S solves the issue and allows to obtain the following productions: $p_1 : A \rightarrow S \wr D$, $p_2 : S \rightarrow B \parallel C$, $p_3 : B \rightarrow \varepsilon$, $p_4 : C \rightarrow \varepsilon$ and $p_5 : D \rightarrow \varepsilon$.

3. Reminder: each task is represented by a grammatical symbol with an attribute named *status* (see definition 2)

the responsibility to the designer to specify it; by default it is a string type. The *process model* (that provides information on the tasks and their sequencing) is provided by the GMWf ($\mathbb{G}$). Note that it is possible, in our case, to represent all forms of routing between tasks (*sequential*, *parallel*, *alternative*[4] and *iterative*[5]).

Compared to other languages used in the literature (*Business Process Model and Notation* [7], *Petri Nets* [8], *Petri Nets with Objects* [3] etc.), ours is mainly distinguished by the fact that it allows designers to statically specify the accreditations of various actors with regard to each task.

3. GMAWfP P2P execution model and corresponding algorithms

3.1. Key elements and constraints of the execution model

3.1.1. The execution environment

To execute a given GMAWfP in a decentralized mode, we use a completely decentralized (P2P) WfMS model (which we call *P2P-WfMS-View*), proposing an artifact-centric execution of business processes specified using grammatical models. For this purpose, instances of a *P2P-WfMS-View* are installed on the sites of the various actors involved in processes execution. These communicate (sending/receiving requests/responses) by service invocation.

During the process execution, each peer keeps a copy of a (global) artifact said to be under execution, which represents the current process execution status. Such an artifact provides information on already executed tasks, on those ready to be executed and on their executors. Technically, an artifact under execution is materialized by the presence within it of *buds*. These indicate at a moment, the only places where contributions are expected. A bud can be either *unlocked* or *locked* depending on whether the corresponding task is ready to be executed or not. Buds are typed; a *bud of type* $X \in \mathcal{S}$ is a leaf node labelled either by $X_{\overline{\omega}}$ or by X_{ω} depending on its state (*locked* or *unlocked*) (see fig. 2). The local actions of a given actor will therefore have the effect of extending its local copy of the (global) artifact by developing, through appropriate productions, the different unlocked buds it contains.

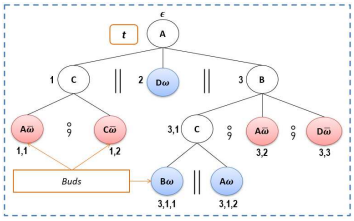

Figure 2. *An intentional representation of an annotated artifact containing buds.*

3.1.2. Confidential execution of certain tasks

For confidentiality reasons, each actor acts potentially on a partial replica of his local copy of the (global) artifact. Technically, a partial replica $t_{\mathcal{V}_i}$ of an artifact t is obtained by

4. Such routing is represented by several productions with the same symbol as left-hand side.
5. To do such routing, all you need is to include recursive symbols in the definition of GMWf productions.

projection (using an operator π said of ***artifact projection***) of t according to the view $\mathcal{V}_i$ of the concerned actor: we note $t_{\mathcal{V}_i} = \pi_{\mathcal{V}_i}(t)$. To ensure the effectiveness of the confidential execution of certain tasks, each actor acts locally through a specialized graphic tool (see fig. 3).

3.1.3. The need of a local GMWf at each site

Since the local actions of a particular actor depend on his perception of the process, it is necessary to control them in order not only to preserve the possible confidentiality of certain tasks, but also to ensure the consistency of local updates. To do this, we must derive a local GMWf on each site, by projecting the global GMWf according to the view of the local actor (***GMWf projection***). This projection is carried out using Π operator and the GMWf obtained is noted $\mathbb{G}_{\mathcal{V}_i} = \Pi_{\mathcal{V}_i}(\mathbb{G})$.

3.1.4. The expansion operation

Still with the aim of ensuring system convergence, the contributions made by a given actor and contained in an updated partial replica $t^{maj}_{\mathcal{V}_i}$, must be integrated into the local copy of the (global) artifact before any synchronization between peers. It is therefore necessary to be able to merge these two artifacts, which are based on two different models. We find here, a version of the ***expansion*** problem as formulated in [2].

3.2. Execution model and peer activity

Globally then, before the execution of a given process, peers are configured using its GMAWfP $\left(\mathbb{W}_f = \left(\mathbb{G}, \mathcal{L}_{\mathcal{P}_k}, \mathcal{L}_{\mathcal{A}_k}\right)\right)$. From the global GMWf $\mathbb{G}$ and the view $\mathcal{V}_i$ of the local actor, each peer derives a local GMWf $\mathbb{G}_{\mathcal{V}_i} = \Pi_{\mathcal{V}_i}(\mathbb{G})$. Then, the execution of a case of the process is triggered when an artifact t is introduced into the system (on the appropriate peer); this artifact is actually an unlocked bud of the type of the axiom A (initial task) of $\mathbb{G}$. During execution, peers synchronize themselves by exchanging their local copies of the artifact being executed.

After receiving an artifact on a given peer, the latter merges it (see fig. 3) with its local copy (if it exists) and then the result is projected. The partial replica obtained is then completed when needed, using the specialized editor.

At the end of the completion, the expansion-pruning of the obtained updated partial replica is made in order to obtain the updated configuration of the (global) artifact local copy. If the resulting configuration shows that the process should be continued at other sites [6], then replicas of the artifact are sent to them by invoking the service "*forwardTo*". Else [7], a replica is returned to the pair from whom the artifact was previously received by invoking the service "*returnTo*".

3.3. The main algorithms of the execution model

The GMAWfP execution model is mainly based on three algorithms: *artifact projection*, *GMWf projection* and *expansion*. In this section, we propose versions of these algorithms.

---

6. This is the case when the artifact contains buds created on the current peer and whose actors accredited in writing are on distant peers.

7. The artifact is complete (it no longer contains buds), or semi-complete (it contains buds that were created on other peers and on which, the actor on the current peer is not accredited in writing).

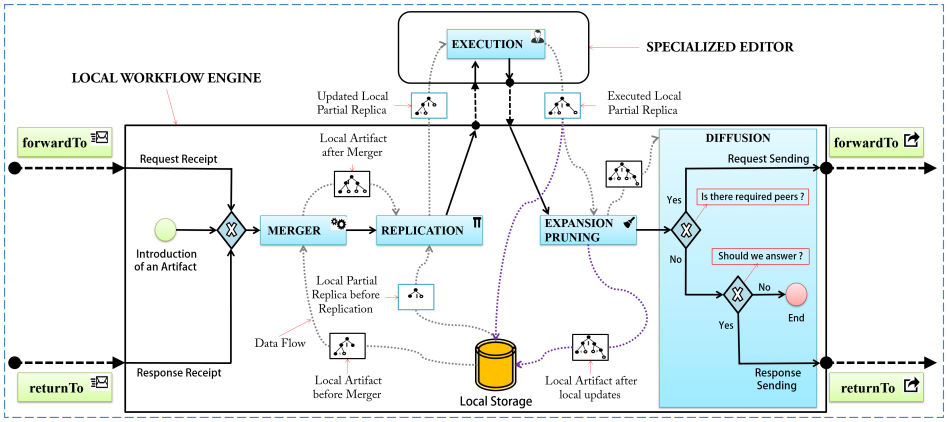

Figure 3. *Activity of a peer in the system.*

3.3.1. The artifact projection algorithm

Technically, the projection $t_{\mathcal{V}_i}$ of an artifact t in the view $\mathcal{V}_i = \mathcal{A}_{A_i(r)}$ is obtained by deleting in t all nodes whose types do not belong to $\mathcal{V}_i$. In our case, the main challenges in this operation are: **(1)** to preserve the previously existing *execution order* between nodes of t, **(2)** to preserve the use of the only two forms of production retained for GMWf and **(3)** to obtain only one artifact after projection to ensure the continuation of processes execution.

Our operation is noted π. Inspired by the one proposed in [2], we project an artifact by preserving the hierarchy (father-son relationship) between nodes of the artifact (we thus meet challenge **(1)**); but in addition, we insert into the projected artifact when necessary, new additional *(re)structuring symbols* (accessible in reading and writing by the actor for whom the projection is made). This enables us to meet challenge **(2)**. Figure 4 presents

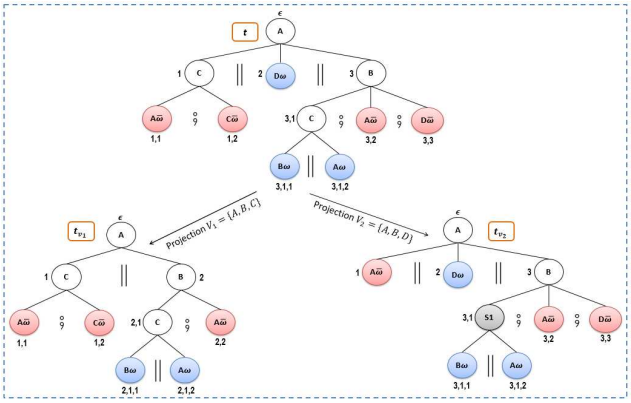

Figure 4. *Example of projections made on an artifact and partial replicas obtained.*

an artifact t as well as two partial replicas t_{v_1} and t_{v_2} respectively obtained from the views $\mathcal{V}_1 = \{A, B, C\}$ and $\mathcal{V}_2 = \{A, B, D\}$. Note the presence in t_{v_2} of the (re)structuring symbol $S1$.

The algorithm: concretely, to project an artifact t according to a given view $\mathcal{V}$ (i.e to find $projs_t = \pi_{\mathcal{V}}(t)$), the following recursive processing is applied to the root node $n = X[t_1, \ldots, t_m]$ (this notation indicates that n is labelled with the symbol X and has m sub-

artifacts $t_1,\ldots,t_m$) of t (note by p_n, the production of the GMWf that was used to extend node n; note also that the algorithm described here can return several artifacts):

Algorithm for projecting a given artifact according to a given view.

- **If symbol X is visible ($X \in \mathcal{V}$)** then :

1. n is kept in the artifact;

2. For each sub-artifact t_i of n, having node $n_i = X_i\left[t_{i_1},\ldots,t_{i_k}\right]$ as root (of which p_{n_i} is the production that was used to extend it), the following processing is applied :

a. The projection of t_i according to $\mathcal{V}$ is done. We obtain the list $proj s_{t_i} = \pi_{\mathcal{V}}(t_i) = \left\{t_{i_{\mathcal{V}_1}},\ldots,t_{i_{\mathcal{V}_l}}\right\}$;

b. If the type of p_{n_i} is the same as the type of p_n or the projection of t_i has produced no more than one artifact ($|proj s_{t_i}| \leq 1$), we just replace t_i by artifacts $t_{i_{\mathcal{V}_1}},\ldots,t_{i_{\mathcal{V}_l}}$ of the list $proj s_{t_i}$;

Otherwise, a new (re)structuring symbol S_i is introduced and we replace the sub-artifact t_i with a new artifact *new_*t_i whose root node is $n_{t_i} = S_i\left[t_{i_{\mathcal{V}_1}},\ldots,t_{i_{\mathcal{V}_l}}\right]$;

3. If the list of new sub-artifacts of n contains only one element t_1 having $n_1 = S_1\left[t_{1_{\mathcal{V}_1}},\ldots,t_{1_{\mathcal{V}_l}}\right]$ (with S_1 a newly created (re)structuring symbol) as root node, we replace in this one, t_1 by the sub-artifacts $t_{1_{\mathcal{V}_1}},\ldots,t_{1_{\mathcal{V}_l}}$ of n_1. This removes a non-important (re)structuring symbol.

- **Else**, n is deleted and the result of the projection ($proj s_t$) is the union of the projections of each of its sub-artifacts: $proj s_t = \pi_{\mathcal{V}}(t) = \bigcup_{i=1}^{m} \pi_{\mathcal{V}}(t_i)$

To avoid that the previous algorithm produces a forest in some cases and thus meet challenge **(3)**, we make the following assumption: *GMAWfP manipulated in this work are such that all actors are accredited in reading on the GMWf axiom (**axiom's visibility assumption**).* It should be noted that with our language, all administrative processes can be designed in a form that validates this assumption.

3.3.2. GMWf projection algorithm

The goal of this algorithm is to derive by projection of a given GMWf $\mathbb{G} = (\mathcal{S},\mathcal{P},A_{\mathbb{G}})$ according to a view $\mathcal{V}$, a local GMWf $\mathbb{G}_{\mathcal{V}} = \left(\mathcal{S}_{\mathcal{V}},\mathcal{P}_{\mathcal{V}},A_{\mathbb{G}_{\mathcal{V}}}\right)$ (we note $\mathbb{G}_{\mathcal{V}} = \Pi_{\mathcal{V}}(\mathbb{G})$). The algorithm we propose is as follows:

Algorithm for projecting a given GMWf according to a given view.

1. First of all, it is necessary to generate all the target artifacts denoted by $\mathbb{G}$; we thus obtain a set $arts_{\mathbb{G}} = \{t_1,\ldots,t_n\}$;

2. Then, each of the target artifacts must be projected according to $\mathcal{V}$. We thus obtain a set $arts_{\mathbb{G}_{\mathcal{V}}} = \left\{t_{\mathcal{V}_1},\ldots,t_{\mathcal{V}_m}\right\}$ (with $m \leq n$ because there may be duplicates; in this case, only one copy is kept) of artifacts partial replicas;

3. Then, collect the different (re)structuring symbols appearing in artifacts of $arts_{\mathbb{G}_{\mathcal{V}}}$, making sure to remove duplicates and to consequently update the artifacts and the set $arts_{\mathbb{G}_{\mathcal{V}}}$. We thus obtain a set $\mathcal{S}_{\mathcal{V}_{Struc}}$ of symbols and a final set $arts_{\mathbb{G}_{\mathcal{V}}} = \left\{t_{\mathcal{V}_1},\ldots,t_{\mathcal{V}_l}\right\}$ (with $l \leq m$) of artifacts. These are exactly the only ones that must be conform to the searched GMWf $\mathbb{G}_{\mathcal{V}}$. So we call them, *local target artifacts for the view* $\mathcal{V}$;

4. At this stage, it is time to collect all the productions that made it possible to build each of the *local target artifacts for the view* $\mathcal{V}$. We obtain a set $\mathcal{P}_{\mathcal{V}}$ of distinct produc-

tions.
The searched local GMWf $\mathbb{G}_{\mathcal{V}} = \left(\mathcal{S}_{\mathcal{V}}, \mathcal{P}_{\mathcal{V}}, A_{\mathbb{G}_{\mathcal{V}}}\right)$ is such as:

a. its set of symbols is $\mathcal{S}_{\mathcal{V}} = \mathcal{V} \cup \mathcal{S}_{\mathcal{V}_{Struc}}$;

b. its set of productions is $\mathcal{P}_{\mathcal{V}}$;

c. its axiom is $A_{\mathbb{G}_{\mathcal{V}}} = A_{\mathbb{G}}$

Illustration: Figure 5 illustrates the research of a local model $\mathbb{G}_{\mathcal{V}_{EC}}$ such as $\mathbb{G}_{\mathcal{V}_{EC}} = \Pi_{\mathcal{V}_{EC}}(\mathbb{G})$ with $\mathcal{V}_{EC} = \mathcal{A}_{EC(r)} = \{A, B, C, D, H1, H2, I1, I2, F\}$. Target artifacts generated from $\mathbb{G}$ (fig. 5(b)) are projected to obtain two *local target artifacts for the view* $\mathcal{V}_{EC}$ (fig. 5(c)). From the local target artifacts thus obtained, the searched GMWf is produced (fig. 5(d)).

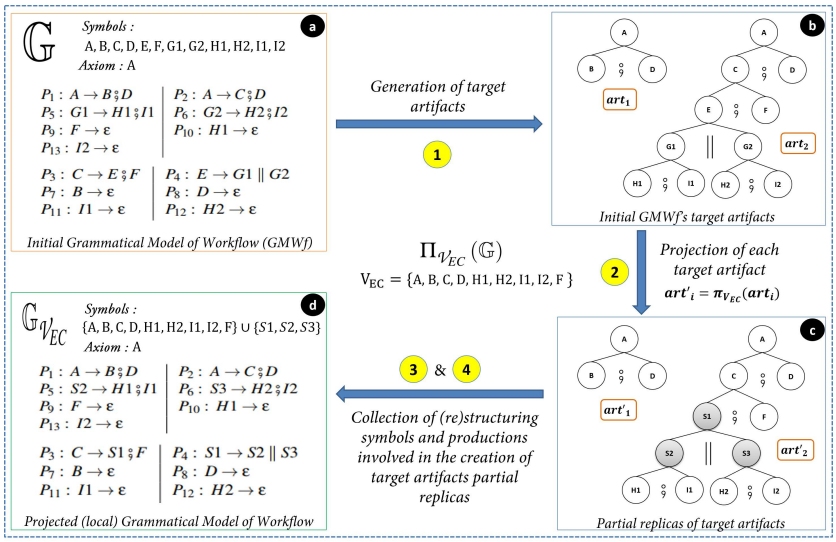

Figure 5. *Example of projection of a GMWf according to a given view.*

REMARK. — The GMWf projection algorithm presented here only works for GMWf that do not allow recursive symbols [8]. We therefore assume that, for this execution model, *the manipulated GMAWfP are those whose GMWf do not contain recursive symbols (**non-recursive GMWf assumption**)*. Therefore, it is no longer possible to express iterative routing between process tasks (in the general case); except in cases where the exact number of iterations is known in advance.

3.3.3. The expansion algorithm

Consider a (global) artifact under execution t and $t_{\mathcal{V}} = \pi_{\mathcal{V}}(t)$, its partial replica on the site of an actor A_i whose view is $\mathcal{V}$. Consider the partial replica $t_{\mathcal{V}}^{maj} \geq t_{\mathcal{V}}$ obtained by developing some unlocked buds of $t_{\mathcal{V}}$ as a result of A_i contribution. The expansion problem is to find a (global) artifact under execution t_f, which integrates nodes of t and $t_{\mathcal{V}}$. To solve this problem made difficult by the fact that t and $t_{\mathcal{V}}$ are conform to two different models ($\mathbb{G}$ and $\mathbb{G}_{\mathcal{V}} = \Pi_{\mathcal{V}}(\mathbb{G})$), we perform a three-way merge [6]. We merge the artifacts t and $t_{\mathcal{V}}$ using a (global) target artifact t_g such that: **(a)** t is a prefix of t_g

8. It is only in this context that all the target artifacts can be enumerated.

($t \leq t_g$) and **(b)** $t_{\mathcal{V}}^{maj}$ is a prefix of the partial replica of t_g according to $\mathcal{V}$ ($t_{\mathcal{V}}^{maj} \leq \pi_{\mathcal{V}}(t_g)$). Our algorithm proceeds in two steps.

Step 1 - Search for the merging guide t_g: the algorithm is as follows :

Algorithm searching for a merging guide.

1. First of all, we have to generate the set $arts_{\mathbb{G}} = \{t_1, \ldots, t_n\}$ of target artifacts denoted by $\mathbb{G}$;
2. Then, we must filter this set to retain only the artifacts t_i admitting t as a prefix (criterion **(a)**) and whose projections according to $\mathcal{V}$ ($t_{i_{\mathcal{V}_j}}$) admit $t_{\mathcal{V}}^{maj}$ as a prefix (criterion **(b)**).
We obtain the set $guides = \{t_{g_1}, \ldots, t_{g_k}\}$ of artifacts that can guide the merging;
3. Finally, we randomly select an element t_g from the set *guides*.

Step 2 - Merging t**,** $t_{\mathcal{V}}^{maj}$ **and** t_g: we want to find an artifact t_f that includes all the contributions already made during the workflow execution. The structure of the searched artifact t_f is the same as that of t_g: hence the interest to use t_g as a guide. The merging is carried out as follows :

Three-way merging algorithm.

A prefixed depth path of the three artifacts is made simultaneously until there is no longer a node to visit in t_g. Let n_{t_i} (resp. $n_{t_{\mathcal{V}_j}^{maj}}$ and $n_{t_{g_k}}$) be the node located at the address w_i (resp. w_j and w_k) of t (resp. $t_{\mathcal{V}}^{maj}$ and t_g) and currently being visited. If nodes n_{t_i}, $n_{t_{\mathcal{V}_j}^{maj}}$ and $n_{t_{g_k}}$ are such that (**processing**):

1. $n_{t_{\mathcal{V}_j}^{maj}}$ is associated with a (re)structuring symbol (fig. 6(d)) then: we take a step forward in the depth path of $t_{\mathcal{V}}^{maj}$ and we resume processing;
2. n_{t_i}, $n_{t_{\mathcal{V}_j}^{maj}}$ and $n_{t_{g_k}}$ exist and are all associated with the same symbol X (fig. 6(a) and 6(b)) then: we insert $n_{t_{\mathcal{V}_j}^{maj}}$ (it is the most up-to-date node) into t_f at the address w_k; if $n_{t_{\mathcal{V}_j}^{maj}}$ is a bud then we prune (delete sub-artifacts) t_g at the address w_k; we take a step forward in the depth path of the three artifacts and we resume processing.
3. n_{t_i}, $n_{t_{\mathcal{V}_j}^{maj}}$ and $n_{t_{g_k}}$ exist and are respectively associated with symbols X_i, X_j and X_k such that $X_k \neq X_i$ and $X_k \neq X_j$ (fig. 6(e)) then: we add $n_{t_{g_k}}$ in t_f at address w_k. This is an upstair bud; we take a step forward in the depth path of t_g and we resume processing.
4. n_{t_i} (resp. $n_{t_{\mathcal{V}_j}^{maj}}$) and $n_{t_{g_k}}$ exist and are associated with the same symbol X (fig. 6(c) and 6(f)) then: we insert n_{t_i} (resp. $n_{t_{\mathcal{V}_j}^{maj}}$) into t_f at the address w_k; if n_{t_i} (resp. $n_{t_{\mathcal{V}_j}^{maj}}$) is a bud, we prune t_g at the address w_k; we take a step forward in the depth path of the artifacts t (resp. $t_{\mathcal{V}}^{maj}$) and t_g, then we resume processing.

Note that an upstair bud is a node associated with a process task that has not yet been executed, and which is revealed during expansion rather than during the execution phase via the specialized editor (this is due to the fact that it is associated with a symbol that does not belong to the view of the considered site). Such a bud appears above other buds in the expansion artifact (hence its name).

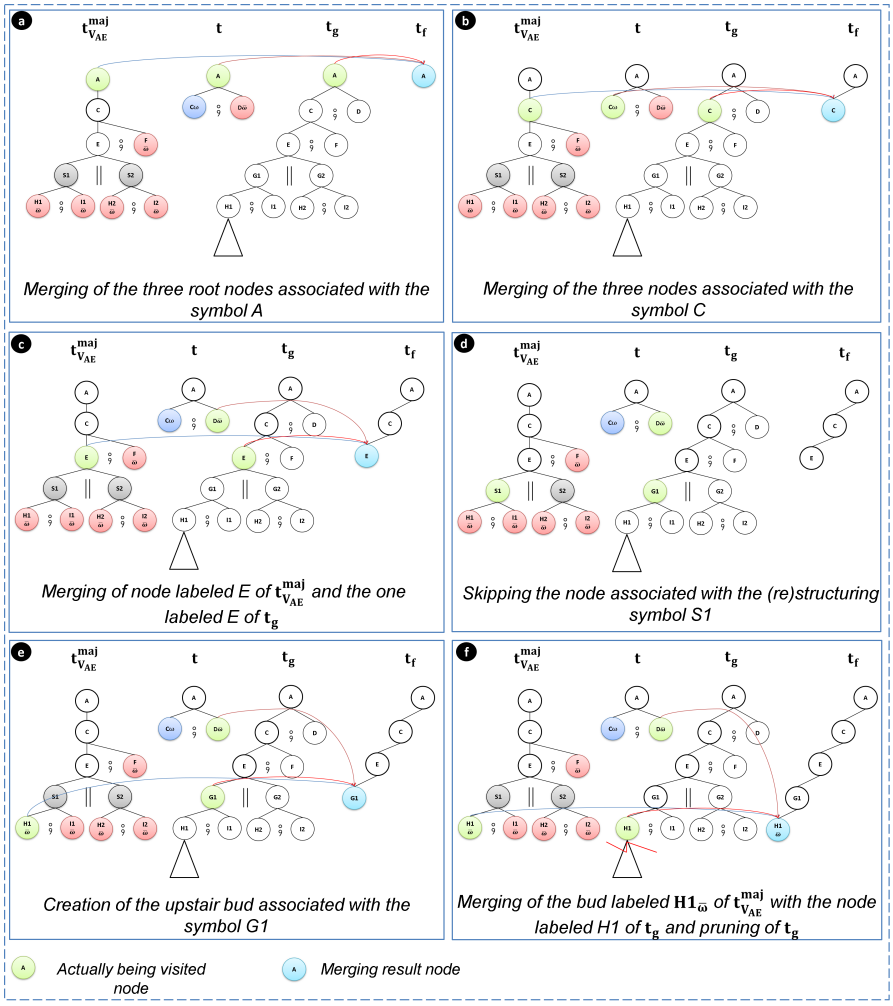

Figure 6. *Some scenarios to be managed during the expansion.*

4. Conclusion

In this work, we proposed LSAWfP, a Workflow Specification Language using grammatical models. We then presented a decentralized and artifact-centric model of workflow processes execution (*P2P-WfMS-View*). Based on the principles of this model, we proposed versions of its key algorithms (*artifact projection*, *GMWf projection* and *expansion-pruning*). These algorithms are perfectly usable (proof of their stability is given in appendix A). We implemented them in several languages and tested them with highly satisfactory results (see appendix B). However, in order to ensure that our algorithms produce the expected results, we have made some assumptions. In particular, the assumption of non-recursivity of GMWf, which had the direct effect of slightly limiting the expressiveness of our language. Therefore, one perspective of this work, is to propose other versions of the algorithms presented here, which would incorporate the same key principles while addressing the assumption of non-recursivity of GMWf in order to offer more comfort to GMAWfP's designers.

References

[1] Wfmc Standards: the Workflow Reference Model, Version 1.1. http://www.aiim.org/wfmc/mainframe.htm, 1995.

[2] Eric Badouel and Maurice Tchoupé Tchendji. Merging Hierarchically-Structured Documents in Workflow Systems. *Electronic Notes in Theoretical Computer Science*, 203(5):3–24, 2008.

[3] Mohamed Amine Chaâbane, Lotfi Bouzguenda, Rafik Bouaziz, and Faïez Gargouri. Etude comparative de quelques langages de spécification de processus workflow. *Actes des 7èmes journées scientifiques des jeunes chercheurs en Génie Electrique et Informatique (GEI)*, 2007.

[4] Monica Divitini, Chihab Hanachi, and Christophe Sibertin-Blanc. Inter-organizational workflows for enterprise coordination. In *Coordination of Internet agents*, pages 369–398. Springer, 2001.

[5] Ayoub Ait Lahcen and Didier Parigot. A Lightweight Middleware for Developing P2P Applications with Component and Service-Based Principles. In *15th IEEE International Conference on Computational Science and Engineering, CSE 2012, Paphos, Cyprus, December 5-7, 2012*, pages 9–16, 2012.

[6] Tom Mens. A State-of-the-Art Survey on Software Merging. *Journal of IEEE Transactions on Software Engineering*, 28(5):449–462, 2002.

[7] Business Process Model. Notation (BPMN) version 2.0. *OMG Specification, Object Management Group*, pages 22–31, 2011.

[8] Wil M. P. van der Aalst. The Application of Petri Nets to Workflow Management. *Journal of Circuits, Systems, and Computers*, 8(1):21–66, 1998.

A. Some properties of our projection algorithms

Proposition 5 *For all GMAWfP $\mathbb{W}_f = \left(\mathbb{G}, \mathcal{L}_{\mathcal{P}_k}, \mathcal{L}_{\mathcal{A}_k}\right)$ verifying the axiom's visibility assumption, the projection of an artifact t which is conform to its GMWf ($t \therefore \mathbb{G}$) according to a given view $\mathcal{V}$, results in a single artifact $t_{\mathcal{V}} = \pi_{\mathcal{V}}(t)$ (stability property of π).*

Proof. Note that the only case in which the projection of an artifact t according to a view $\mathcal{V}$ produces a forest, is when the root node of t is associated with an invisible symbol X ($X \notin \mathcal{V}$). Knowing that $t \therefore \mathbb{G}$ and that $\mathbb{W}_f$ validates the axiom's visibility assumption, it is deduced that the root node of t is labelled by the axiom $A_{\mathbb{G}}$ of $\mathbb{G}$ and that $A_{\mathbb{G}} \in \mathcal{V}$ (hence the uniqueness of the produced tree). Since the projection operation preserves the form of productions, it is concluded that $t_{\mathcal{V}} = \pi_{\mathcal{V}}(t)$ is an artifact. □

Proposition 6 *For all GMAWfP $\mathbb{W}_f = \left(\mathbb{G}, \mathcal{L}_{\mathcal{P}_k}, \mathcal{L}_{\mathcal{A}_k}\right)$ verifying the axiom's visibility and the non-recursivity of GMWf assumptions, the projection of its GMWf $\mathbb{G} = (\mathcal{S}, \mathcal{P}, A_{\mathbb{G}})$ according to a given view $\mathcal{V}$, is a GMWf $\mathbb{G}_{\mathcal{V}} = \Pi_{\mathcal{V}}(\mathbb{G})$ for a GMAWfP $\mathbb{W}_{f_{\mathcal{V}}}$ verifying the assumptions of axiom visibility and non-recursivity of GMWf (stability property of Π).*

Proof. As $\mathbb{W}_f = \left(\mathbb{G}, \mathcal{L}_{\mathcal{P}_k}, \mathcal{L}_{\mathcal{A}_k}\right)$ validates the non-recursivity of GMWf assumption, the set of target artifacts ($arts_{\mathbb{G}} = \{t_1, \ldots, t_n\}$) that it denotes is finished and can therefore be fully enumerated. Knowing further that $\mathbb{W}_f$ validates the axiom's visibility assumption, we deduce that the set $arts_{\mathbb{G}_{\mathcal{V}}} = \left\{t_{\mathcal{V}_1} = \pi_{\mathcal{V}}(t_1), \ldots, t_{\mathcal{V}_n} = \pi_{\mathcal{V}}(t_n)\right\}$ is finite and the root node of each artifact $t_{\mathcal{V}_i}$ is associated with the axiom $A_{\mathbb{G}}$ of $\mathbb{G}$ (see proposition 5). $\mathbb{G}_{\mathcal{V}}$

being built from the set $arts_{\mathbb{G}_{\mathcal{V}}}$, its axiom $A_{\mathbb{G}_{\mathcal{V}}} = A_{\mathbb{G}}$ is visible to all actors and its productions are only of the two forms retained for GMWf. In addition, each new (re)structuring symbol ($S \in \mathcal{S}_{\mathcal{V}_{Struc}}$) is created and used only once to replace a symbol that is not visible and not recursive (by assumption) when projecting artifacts of $arts_{\mathbb{G}}$. The new symbols are therefore not recursive. By replacing in $\mathcal{L}_{\mathcal{A}_k}$ the view $\mathcal{V}$ by $\mathcal{V} \cup \mathcal{S}_{\mathcal{V}_{Struc}}$, we obtain a new set $\mathcal{L}_{\mathcal{A}_{\mathcal{V}_k}}$ of accreditations for a new GMAWfP $\mathbb{W}_{f_{\mathcal{V}}} = \left(\mathbb{G}_{\mathcal{V}}, \mathcal{L}_{P_k}, \mathcal{L}_{\mathcal{A}_{\mathcal{V}_k}}\right)$ verifying the assumptions of axiom visibility and non-recursivity of GMWf. □

Proposition 7 *For all GMAWfP $\mathbb{W}_f = \left(\mathbb{G}, \mathcal{L}_{P_k}, \mathcal{L}_{\mathcal{A}_k}\right)$ verifying the axiom's visibility and the non-recursivity of GMWf assumptions, the projection of an artifact t which is conform to the GMWf $\mathbb{G}$ according to a given view $\mathcal{V}$, is an artifact which is conform to the projection of $\mathbb{G}$ according to $\mathcal{V}$ ($\forall t \therefore \mathbb{G}, \pi_{\mathcal{V}}(t) \therefore \Pi_{\mathcal{V}}(\mathbb{G})$).*

Proof. $t \therefore \mathbb{G}$ implies that t is a target artifact of $\mathbb{G}$ ($t \in arts_{\mathbb{G}}$). Consequently $t_{\mathcal{V}} = \pi_{\mathcal{V}}(t)$, through the renaming of some potential (re)structuring symbols, is part of the set $arts_{\mathbb{G}_{\mathcal{V}}}$ of artifacts that have generated $\mathbb{G}_{\mathcal{V}} = \Pi_{\mathcal{V}}(\mathbb{G})$. We conclude that $t_{\mathcal{V}} \therefore \mathbb{G}_{\mathcal{V}}$. □

B. Experimentation with P2PTinyWfMS

The algorithms presented in this manuscript, have been coded in Haskell and Java then tested with satisfactory results. In java, we introduced these algorithms in the tool *P2PTinyWfMS*. It is a tool developed under Eclipse [9] and dedicated to the simulation of the completely decentralized execution of administrative workflows specified using LSAWfP. *P2PTinyWfMS* has a front-end for the display and the graphical editing of artifacts handled during the execution of a business process (see fig. 7), as well as a Service-Oriented communication module built using SON (Shared-data Overlay Network) [5].

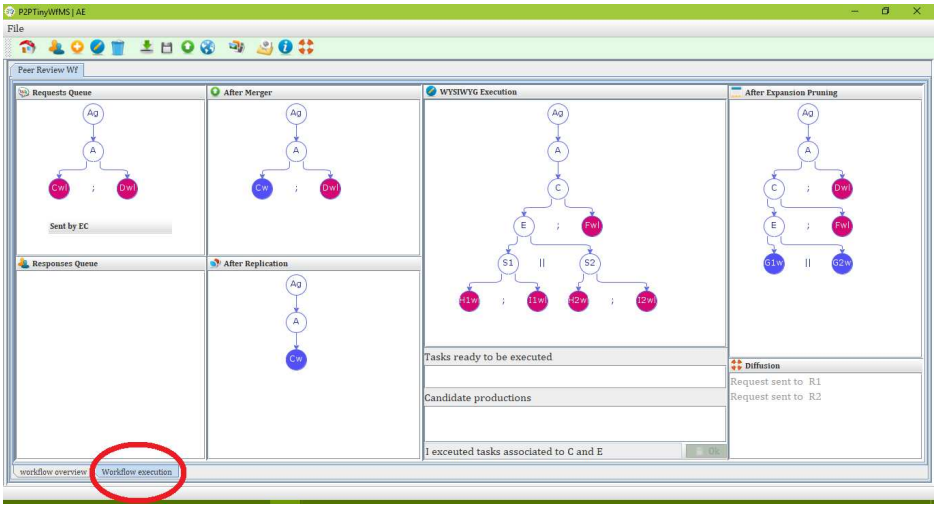

Figure 7. *Simulation of the execution of a process using P2PTinyWfMS.*

9. https://www.eclipse.org